Math 320 Programming Project II - Fall 2018

Please submit all project parts on the Moodle page for MAT 320. You should include all necessary files to recompile, and a working executable, all in a zipped folder (one file for upload). Time-stamp determines the submit time, due by midnight on the due-date.

Part II: Discrete Fourier Transform – Basic and Recursive Forms

Due: Friday, Oct 12, midnight

For each part below, you will need to write a command line program in C++ with text input and output. Programs should compile with g++ on Linux and Clang on Mac, so no Windows specific code is allowed.

1. Discrete Fourier Transform (DFT) - Basic Version
   - input: command line arg N, text file of N complex numbers
   - output: DFT of input as list of N complex numbers
2. Recursive (Fast) Fourier Transform (FFT) - First Version
   - input: command line arg $N = 2^m$, text file of N complex numbers
   - output: DFT of input as list of N complex numbers

Notes:

1. Please follow these naming guidelines: there should be two separate programs, one called dft1.cpp and one called fft1.cpp. They should compile on the command line with g++ and should not require any other headers or linked files.
2. The basic version of the DFT is Formula (2.6) on page 152 of the text book. If the inputs are labelled $x_0, \dots, x_N$ then the outputs are $X_0, \dots, X_N$.

3. The recursive version of the DFT, which is the first version of the FFT, is described in section 6 of chapter 8, pages 162-163 of the text book. The inputs and outputs are identical to the DFT. This time you write a recursive function which is based on Formula (6.9) on page 163.

4. To implement the recursive function I recommend you write a VOID function which takes as input arguments: two arrays of complex doubles, one for input and one for output, and the positive integer $N = 2^m$. Each recursive call will maintain that the integer is still a power of 2. The case when $N = 1$ will be that the function simply writes the complex number input to the output. For general $N = 2^m$ the function should create 4 new arrays of $N/2$ complex numbers, two for the even indices and two for the odd indices. Each pair of arrays is again used for input and output. Next, the inputs are assigned according to evens and odds, then the outputs are computed with the recursive call to the function using arguments: eveninputs, evenoutputs, $N/2$, and then: oddinputs, oddoutputs, $N/2$. Finally, you will need two loops that implement the equation (6.9) which puts the even and odd outputs back together to give the level N outputs as a sum with a special exponential factor in front of the odds. The reason for two loops is that (6.9) is slightly different for the second half of the outputs, where the exponential factor simplifies as described in the paragraph after (6.9).